ALGORITMA KOMPRESI FRAKTAL *SEQUENTIAL* DAN PARALEL UNTUK KOMPRESI CITRA

Satrya N. Ardhytia dan Lely Hiryanto

Laboratorium Pemrosesan Paralel, Fakultas Teknologi Informasi, Universitas Tarumanagara, Kantor Rektorat Kampus I Gedung Utama Lt. 5Jl. Letjen S. Parman No. 1**,** Jakarta Barat, 11440, Indonesia

E-mail: lely@fti.untar.ac.id

Abstrak

Kompresi citra adalah proses mengurangi ukuran dari citra dengan mengurangi kualitas dari citra tersebut. Metode Fraktal yang digunakan bekerja dengan mencari kemiripan pada *piksel-piksel* citra dan mengelompokkannya dalam beberapa *cluster*. Semakin tinggi tingkat kemiripan pada citra, rasio kompresi akan semakin baik. Pada citra berwarna (RGB) metode tersebut diulang sebanyak tiga kali, masing-masing untuk satu elemen warna. Hasil akhir dari proses kompresi adalah tiga virtual *codebook*, masing-masing untuk satu elemen warna, yang menyimpan nilai dari *brightness*, *contrast*, dan tipe transformasi *affine* yang digunakan untuk tiap *cluster*. Proses dekompresi dari metode ini adalah dengan membentuk citra kosong dengan resolusi yang sama dengan citra asli dan mengisikan nilai RGB pada tiap *piksel* yang bersangkutan dengan menghitung nilai yang tersimpan pada virtual *codebook*. Dengan menggunakan nilai *Coefficient of Variation* (CV) sebagai penyesuaian nilai standar deviasi dan 57 citra BMP24-bit, hasil pengujian menunjukkan rasio kompresi rata-rata sebesar 41.79%. Dengan metode paralel yang digunakan, proses kompresi citra berwarna menunjukkan rata-rata nilai *speed-up* sebesar 1.69 dan nilai efisiensi prosesor sebesar 56.34%.

Kata Kunci: *kompresi citra, kompresi fraktal paralel, kompresi fraktal sequential*

Abstract

Image compression is a process of reducing the size of the image by reducing the quality of the image. Fractal method is used to work by searching for similarities in the image pixels, and group them in clusters. The higher the degree of resemblance to the image, the better the compression ratio. In the color image (RGB) the method is repeated three times, each for one color element.The end result of the compression process is a three virtual codebook, each for one color element, which stores the value of the brightness, contrast, and the type of affine transformation are used for each cluster. Decompression process of this method is to form a blank image with the same resolution with the original image and fill in the RGB values at each pixel corresponding to the count value stored in the virtual codebook.By using the Coefficient of Variation (CV) as an adjustment value and standard deviation of 57 pieces of 24-bit BMP images, test results showed an average compression ratio of 41.79%. With the parallel method is used, the compression process of color image shows the average speed-up values of 1.69 and the processor efficiency of 56.34%.

Keywords: *image compression, parallel fractal compression, sequential fractal compression*

1. Pendahuluan

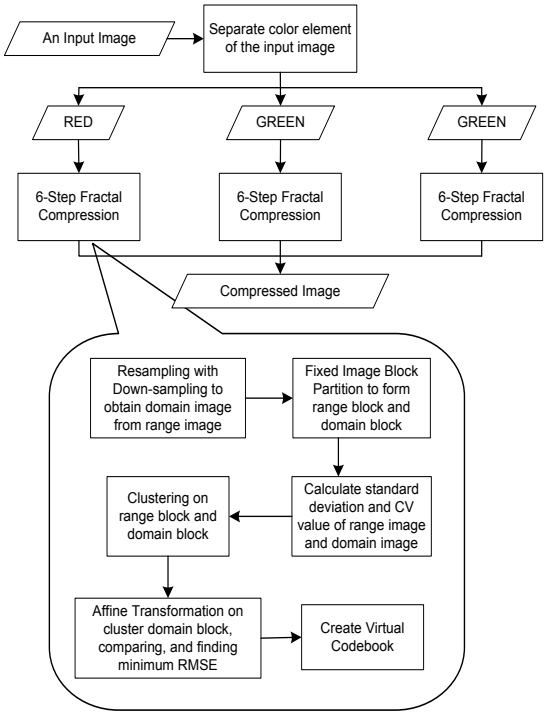
Algoritma Fraktal adalah salah satu dari berbagai metode *lossy image compression* [1]. Metode kompresi ini mengusung fakta bahwa dalam suatu citra ada bagian yang mirip dengan bagian lainnya pada citra tersebut. Gambar 1 menunjukkan cara kerja dari algoritma kompresi Fraktal. Bayangkan ada tiga matriks, masing-masing dengan 1 elemen warna (merah, hijau, dan biru) dari citra *input*, proses kompresi dimulai dengan membuat *domain image* dari *range image* (citra asli). *Domain image* dibentuk dengan melakukan *down-sampling* terhadap *range image*. Terhadap *domain image* dan *range image* dilakukan partisi dengan metode *fixed block partition*. Kemudian proses *clustering* dilakukan terhadap *domain block* dan *range block* yang telah terbentuk menghasilkan *domain block cluster* dan *range block cluster*. Pada setiap *domain block cluster* dilakukan transformasi *affine* dan dibandingkan terhadap *range block cluster* untuk dicari kemiripannya dengan menghitung faktor *brightness*, *contrast*, dan *root mean square error* (RMSE).

Proses tersebut dilakukan untuk mencari nilai RMSE minimum untuk setiap *range block cluster*. Kemudian virtual *codebook* disusun berisikan nilai *brightness*, *contrast*, dan jenis transformasi yang dilakukan untuk setiap *range block cluster* dengan RMSE minimum.

Setiap *piksel-piksel* pada sebuah citra berwarna, memiliki tiga nilai dari setiap elemen warna: [Merah, Hijau, Biru]. Sebagai contoh, sebuah *piksel* RGB [255,0,0] menampilkan warna merah, dan RGB [255,255,0] menampilkan warna kuning. Untuk citra berwarna, proses kompresi dilakukan tiga kali, masing-masing untuk setiap elemen warna dan menghasilkan tiga virtual *codebook*.

Proses dekompresi dari kompresi Fraktal dimulai dengan membentuk sebuah citra kosong dengan resolusi yang sama dengan citra asli dan kemudian dipartisi dengan ukuran blok yang sama dengan citra asli. Setiap blok partisi diisikan nilai dari transformasi *domain* blok yang sesuai setelah dikalikan dengan nilai *contrast* dan dijumlahkan dengan nilai *brightness*, sesuai dengan indeks pada virtual *codebook*.

Seiring dengan peningkatan kualitas citra, ukuran dari citra tersebut juga semakin meningkat. Dalam penyimpanannya harus memperhatikan kapasitas dari media penyimpanan dan ukuran dari citra itu sendiri. Oleh karena itu dilakukan kompresi citra.



Gambar 1. Proses kompresi Fraktal.

Berdasarkan proses kompresi Fraktal secara *sequential*, waktu untuk proses kompresi yang dilakukan pada setiap citra sangat bervariasi. Untuk citra dengan resolusi besar dan variasi warna yang banyak, waktu proses akan meningkat secara signifikan.

Disinilah pemrosesan paralel akan mengurangi waktu pemrosesan citra. Pada dasarnya pemrosesan paralel adalah membagi tugas-tugas yang dapat dikerjakan secara bersamaan kepada sejumlah prosesor. Semakin banyak jumlah prosesor yang dilibatkan, proses pun akan semakin cepat. Pemrosesan paralel yang digunakan adalah *Massage Passing Computing*. Prosesor-prosesor yang digunakan dibagi menjadi sebuah *master* dan beberapa *slave*. Data *input* dari *master* akan dikirimkan atau dibagi-bagi kepada semua *slave* termasuk *master* sendiri untuk diproses lebih lanjut secara bersamaan. Proses paralel ini akan membuat proses kompresi citra menjadi lebih efisien.

Beberapa pendekatan telah diajukan oleh para peneliti seperti pada [2][3] untuk mengurangi waktu pemrosesan dengan menggunakan pemrosesan paralel. Tujuan dari penelitian ini adalah untuk menghasilkan citra dengan ukuran yang lebih kecil sehingga lebih efisien dalam penyimpanan maupun proses transfer. Dengan proses paralel yang diterapkan pada program ini diharapkan waktu kompresi citra akan semakin cepat. Di sini, peneliti tidak hanya membahas kualitas dari hasil kompresi dengan algoritma kompresi Fraktal, tetapi juga memaparkan solusi paralel untuk mendapatkan percepatan untuk proses kompresi.

Citra digital terbagi dua, yaitu citra yang berupa vektor atau raster. Pada umumnya citra digital lebih mengacu pada citra raster. Citra raster merupakan kumpulan dari *piksel-piksel* yang berisi informasi warna yang tersusun dalam baris dan kolom. Nilai baris dan kolom dalam suatu citra disebut resolusi [4].

Tipe dari citra raster dapat diklasifikasikan menurut warnanya sebagai contoh adalah citra binari (*black* & *white*), *grayscale*, dan *color*. Dalam pembuatan program ini, citra yang diproses adalah citra berwarna. Lebih spesifiknya adalah model warna RGB (*Red*, *Green*, dan *Blue*) [4].

Model warna RGB adalah sebuah model warna yang berdasarkan pada penjumlahan dari tiga warna dasar yaitu, merah, hijau, dan biru. Dalam sebuah citra RGB, sebuah *piksel* memuat nilai dari tiga warna dasar tersebut. Dalam sebuah citra RGB 24-bit, nilai RGB masing-masing

berkisar antara 0-255. Sebagai contoh representasi warna pada sebuah *piksel* antara lain, RGB [255,0,0] menghasilkan warna merah, RGB [0,255,0] menghasilkan warna biru, RGB [0,0,255] menghasilkan warna hijau, RGB [255,0,255] menghasilkan warna magenta, RGB [255,255,0] menghasilkan warna kuning, RGB [0,255,255] menghasilkan warna sian, RGB [0,0,0] menghasilkan warna hitam, RGB [255,255,255] menghasilkan warna putih.

2. Metodologi

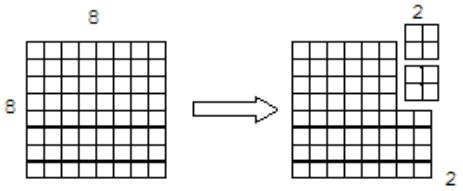
Kompresi citra merupakan implementasi kompresi data pada citra digital. Tujuan dari kompresi citra adalah untuk mengurangi *redundancy* data sebuah citra agar dapat mengurangi ukuran *byte* dari sebuah citra. Kompresi citra dibagi menjadi kompresi *lossless* dan kompresi *lossy*. Kompresi *lossless* tidak akan mengurangi kualitas citra saat didekompresi, sedangkan kompresi *lossy* akan menghasilkan artifak, terlebih pada saat rasio kompresi tinggi. Kompresi citra dengan metode Fraktal ini termasuk kompresi *lossy* [5].

Secara umum, fraktal dikenal sebagai suatu bentuk geometri yang dapat dipecah-pecah menjadi beberapa bagian, di mana bagian-bagian tersebut merupakan bentuk yang sebelumnya dengan ukuran yang lebih kecil. Bentuk geometri ini pertama kali dikemukakan oleh Benoît Mandelbrot pada tahun 1975. Barnsley dan Hurd menggunakan konsep Fraktal ini untuk mengemukakan suatu pendekatan kompresi baru [1]. Kompresi fraktal mengusung fakta bahwa dalam sebuah citra ada suatu bagian yang mirip dengan bagian lain dari citra tersebut [1-3]. Algoritma Fraktal mengubah bagian-bagian yang mirip tersebut menjadi data metematis yang disebut "kode Fraktal" yang akan digunakan untuk membentuk ulang citra yang dikompresi. Citra yang dikompresi dengan metode Fraktal akan kehilangan resolusinya, sehingga memungkinkan untuk membentuk kembali citra tersebut dalam resolusi yang berbeda tanpa adanya artifak. Gambar 1 menunjukkan cara kerja dari proses kompresi Fraktal.

Proses kompresi citra dimulai dari pembuatan *domain image* dari citra *input* (*range image*). *Domain image* didapat dari proses *down sampling* pada *range image*. Kemudian pada *domain image* dan *range image* dilakukan proses partisi dengan metode *fixed partition block* untuk mendapatkan *domain block* dan *range block*. Pada *domain block* dan *range block* yang terbentuk dilakukan *clustering* dan dilakukan perbandingan kemiripan dari tiap *cluster range block* terhadap *cluster domain block* yang telah dilakukan transformasi *affine* dengan menghitung faktor *contrast*, *brightness*, dan *root mean square error* (RMSE). Proses ini dilakukan untuk mencari nilai RMSE minimum pada tiap *cluster range block*. Setelah didapat nilai RMSE minimum, selanjutnya dibentuk virtual *codebook* yang berisi nilai faktor *contrast*, *brightness*, indeks dari *cluster domain block* dan koefisien transformasi.

Dengan mengacu pada gambar 1, ada enam langkah untuk melakukan kompresi citra dengan metode Fraktal [1]. Pertama, *resampling* adalah sebuah proses untuk mengubah dimensi dari suatu citra digital. Ada dua jenis dari *resampling*, yaitu *down-sampling* untuk mengurangi ukuran, dan *up-sampling* untuk menambah ukuran [1]. Peneliti menggunakan *down-sampling* sebagai langkah pertama dari proses kompresi. Proses *down-sampling* dilakukan dengan menghitung nilai rata-rata dari blok 2×2 *piksel* sebagai nilai dari *piksel* yang bersangkutan pada citra hasil. Sebagai contoh, proses *down-sampling* pada citra berukuran 256×256 *piksel* akan menghasilkan citra dengan ukuran 128×128 *piksel*.

Langkah kedua, partisi berarti memecah atau memisahkan suatu objek menjadi bagian-bagian [6]. Jenis partisi yang digunakan di sini adalah *fixed block partition*. Dengan metode ini, citra dipecah-pecah menjadi bagian berbentuk persegi dengan ukuran yang sama. Ukuran blok yang digunakan di sini adalah blok 4×4 *piksel*, di mana citra dengan ukuran 256×256 *piksel* akan menghasilkan 4096 blok berukuran 4×4 *piksel*. Ilustrasi untuk proses partisi dapat dilihat pada gambar 2.



Gambar 2. *Fixed image block partition of 4x4 block* [7].

Langkah ketiga, *Coefficient of Variation*. Setelah proses partisi selesai, nilai standar deviasi pada masing-masing *range image* dan *domain image* dihitung. Standar deviasi adalah nilai perbedaan dari objek yang dibandingkan. Semakin besar nilai standar deviasi berarti perbedaan semakin besar [8]. Persamaan 1 digunakan untuk menghitung nilai standar deviasi.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad , \quad \bar{x} = \frac{x_1 + x_2 + \cdots x_n}{n}$$

$$\sigma = \sqrt{\frac{1}{n-1} + \sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad (1)$$

Keterangan:
Σ = *root mean square deviation*
n = jumlah sampel
x_i = nilai sampel ke-i
x̄ = nilai rata-rata sampel

Nilai dari standar deviasi kemudian digunakan untuk menghitung nilai *Coefficient of Variation* (CV). CV adalah rasio nilai standar deviasi dengan nilai rata-rata sampel [9]. Persamaan 2 menunjukkan perhitungan nilai CV.

$$cv = \frac{\sigma}{\bar{x}} \mathrm{X}\, 100\% \qquad (2)$$

Keterangan:
CV = *Coefficient of variation*
σ = *root mean square deviation*
$\bar{x}$ = nilai rata-rata sampel

Langkah keempat, *clustering*. Setelah semua proses di atas selesai, terhadap *range image* dan *domain image* dilakukan *clustering*. *Clustering* adalah suatu proses untuk mengelompokkan berbagai objek ke dalam *cluster-cluster* berdasarkan kemiripan. Algoritma *clustering* yang digunakan adalah *subtractive clustering*. Metode ini melakukan *clustering* dengan membandingkan setiap objek dengan objek setelahnya sampai semua objek selesai dibandingkan. Metode perbandingan yang digunakan adalah *root mean square error* (RMSE) [6][10]. Dengan menggunakan *clustering*, tingkat kemiripan antar blok citra dihitung, dan dikelompokkan dalam *cluster-cluster* sesuai dengan tingkat kemiripannya. Tingkat kemiripan tiap blok dihitung berdasarkan nilai *contrast* dan *brightness*, menggunakan persamaan 3 dan 4 [9].

$$S_i = \frac{[m\sum_{i=1}^{m} D_i R_i - \sum_{i=1}^{m} D_i \sum_{i=1}^{m} R_i]}{[m\sum_{i=1}^{m}(D_i)^2 - (\sum_{i=1}^{m} D_i)^2]} \qquad (3)$$

$$O_i = \frac{1}{m}\left[\sum_{i=1}^{m} R_i - S_i \sum_{i=1}^{m} D_i\right] \qquad (4)$$

$$RMSE^2 = \frac{1}{m}[\sum_{i=1}^{m} R_i^2 + S_i\,(S_i \sum_{i=1}^{m} D_i^2 - 2\sum_{i=1}^{m} D_i R_i + 2\,O_i \sum_{i=1}^{m} D_i) + O_i(mO_i - 2\sum_{i=1}^{m} R_i)] \qquad (5)$$

Keterangan:
R_i = *rangeblock* ke-i
D_(i) = *domain block transformation* ke-i
m = jumlah *piksel* dalam satu *block*
S_i = faktor *contrast block* ke-i
O_i = faktor *brightness block* ke-i
RMSE = *root mean square error*

Nilai RMSE (persamaan 5) yang didapat kemudian dibandingkan dengan nilai standar deviasi. Jika nilai RMSE lebih kecil atau sama dengan nilai standar deviasi, blok yang dibandingkan dikelompokkan dalam satu *cluster*, kemudian nilai *piksel* dari *cluster* tersebut adalah nilai rata-rata dari seluruh blok dalam *cluster* tersebut. Selama proses ini berlangsung, tabel indeks juga dibuat untuk menentukan lokasi blok yang bersangkutan untuk kepentingan proses dekompresi.

Langkah kelima adalah melakukan transformasi *affine* untuk setiap domain *cluster*. Transformasi *affine* adalah kumpulan dari berbagai macam transformasi linear. Transformasi yang digunakan antara lain refleksi terhadap x dan y, refleksi terhadap garis x-y dan x=y, rotasi 0°, 90°, 180°, dan 270°. Hasil transformasi kemudian dibandingkan terhadap *rangecluster* dan dihitung kembali faktor *contrast, brightness* dan RMSE. Jenis transformasi dengan nilai RMSE paling rendah disimpan bersama nilai *brightness, contrast*, dan jenis transformasi yang digunakan. Proses ini dilakukan untuk semua domain *cluster*.

Langkah keenam, *creating virtual codebook*. Faktor *contrast* (S), faktor *brightness* (O), nilai RMSE dan jenis transformasi yang digunakan disimpan dalam virtual *codebook*. Nilai RMSE dalam setiap *codebook* akan dibandingkan untuk mencari nilai RMSE minimun yang disimpan dalam virtual *codebook* final. Langkah 1 sampai 6 dilakukan untuk setiap elemen warna dimulai dari merah, hijau, dan terakhir biru menghasilkan tiga virtual *codebook* final. Setelah semua elemen warna selesai diproses, langkah terakhir adalah menulis ketiga virtual *codebook* tersebut ke dalam *file* kompresi dimulai dari elemen merah, hijau, dan terakhir biru.

Untuk proses dekompresi, tahapan dimulai dengan membuat layar citra baru dengan resolusi yang sama dengan citra asli. Layar citra baru kemudian dipartisi dengan blok partisi yang sama dengan citra asli. Kemudian setiap blok partisi tersebut diisi dengan nilai dari transformasi *domain block* yang terbaik dikalikan dengan faktor *contrast* kemudian dijumlahkan dengan faktor *brightness* dan disusun berdasarkan indeks yang berada dalam virtual *codebook*.

Untuk menunjukkan kompleksitas waktu dari langkah-langkah di atas, peneliti menggunakan notasi Big-Oh. Berikut ini adalah hasil analisis dengan mengasumsikan variabel n sebagai panjang dan lebar dari citra *input*-nya:

Pertama adalah *resampling*:

$$n \times n = n^2 \rightarrow O(n^2)$$

Kedua adalah partisi blok:

$$n^2 + \left(\frac{n}{2} \times \frac{n}{2}\right) = n^2 + \frac{n^2}{4} \rightarrow O(n^2)$$

Ketiga adalah standar deviasi:

$$n^2 + \frac{n^2}{4} \rightarrow O(n^2)$$

Keempat adalah *range clustering*:

$$\frac{\frac{n^2}{16}\left(\frac{n^2}{16}+1\right)}{2} = \frac{\left(\frac{n^4}{256}+\frac{n^2}{16}\right)}{2} = \frac{n^4}{512}+\frac{n^2}{32} \rightarrow O(n^4)$$

Kelima adalah *domain clustering*:

$$\frac{\frac{n^2}{64}\left(\frac{n^2}{64}+1\right)}{2} = \frac{\left(\frac{n^4}{4096}+\frac{n^2}{64}\right)}{2} = \left(\frac{n^4}{4096}+\frac{n^2}{128}\right) \rightarrow O(n^4)$$

Keenam adalah VCB:

$$\left(\frac{n^2}{16} \times 7\right) \times \frac{n^2}{64} = \frac{7n^4}{1024} \rightarrow O(n^4)$$

Ketujuh adalah *file writing*:

$$\frac{n^2}{16} + \frac{n^2}{16} = \frac{n^2}{8} \rightarrow O(n^4)$$

Setiap proses tersebut diulang sebanyak tiga kali, menghasilkan kompleksitas total untuk algoritma *sequential* sebesar $3 \times (n^2 + n^2 + n^2 + n^4 + n^4 + n^4 + n^2) = O(n^4)$.

Pada algoritma Fraktal *sequential* (bagian 2), keenam langkah tersebut diulang sebanyak tiga kali masing-masing untuk setiap elemen warna. Oleh karena itu, peneliti mengajukan algoritma Fraktal yang memisahkan proses tersebut agar dikerjakan secara terpisah untuk masing-masing elemen warna.

Dengan mengacu pada gambar 1, peneliti hanya membutuhkan tiga unit prosesor, yang masing-masing mengerjakan satu elemen warna. Gambar 3 mengilustrasikan pemetaan proses paralel untuk setiap prosesor dengan lebih detil. Saat program menulis *file fractal* (*file* citra hasil kompresi), setiap komputer menulis virtual *codebook* satu persatu dimulai dari elemen warna pertama, yaitu merah, kemudian hijau, dan terakhir biru ke dalam satu *file*.

Untuk menghitung kinerja dari algoritma Fraktal paralel yang diajukan, peneliti mempertimbangkan dua faktor, yaitu faktor *speed-up* dan nilai efisiensi dari setiap prosesor, di mana persamaan untuk menghitung kedua nilai tersebut adalah sebagai berikut:

$$S(p) = \frac{t_s}{t_p} \quad (7)$$

Keterangan:
S(p) = nilai *speed-up*
t_s = waktu pemrosesan *sequential*
t_p = waktu pemrosesan paralel dengan p prosesor

$$E = \frac{t_s}{t_p \times p} \times 100\% \quad (8)$$

Keterangan:
E = nilai efisiensi dalam persen
t_s = waktu pemrosesan *sequential*
t_p = waktu pemrosesan paralel dengan p prosesor
p = jumlah prosesor yang digunakan

Karena enam langkah proses kompresi untuk elemen warna merah, hijau, dan biru dilakukan terpisah pada tiga komputer yang berbeda, kompleksitas algoritma paralel dipisahkan menjadi waktu komunikasi dan waktu komputasi. Dengan mengasumsikan komunikasi data antar prosesor hanya terjadi saat proses penulisan virtual *codebook* ke dalam *file* kompresi, hasil kompleksitas akhir adalah$(n^2 + n^2 + n^2 + n^4 + n^4 + n^4 + n^2) = O(n^4)$ untuk waktu komputasi ditambah $O(n^2)$ untuk waktu komunikasi. Sehingga total kompleksitas algoritma paralel adalah $O(n^4)$.

Meskipun kompleksitas waktu paralel dan *sequential* sama, dengan analisis yang lebih detil dapat dilihat bahwa waktu komputasi paralel akan tiga kali lebih cepat dari pada *sequential*. Dengan memperhitungkan waktu komunikasi antara tiga prosesor, dalam implementasinya sangat sulit untuk mencapai *speed-up* sampai tiga kali. Namun karena waktu komunikasi tidak akan terlalu signifikan dibanding waktu komputasi, peneliti yakin *speed-up* akan tetap didapat. Hal tersebut telah dibuktikan dari hasil pengujian pada bagian berikutnya.

3. Hasil dan Pembahasan

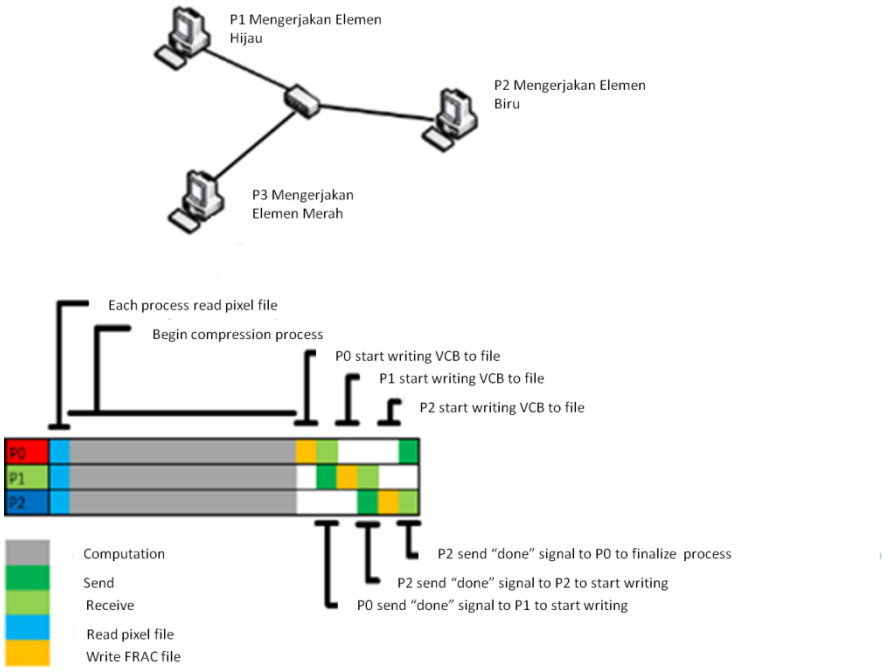
Program kompresi Fraktal ini dibuat dengan menggunakan Microsoft Visual C/C++ dan MPICH NT versi 1.2 untuk Microsoft Windows yang dirancang oleh Argonne National Laboratory [11]. Program ini dijalankan dalam sebuah *cluster* komputer.

Computer cluster adalah kelompok dari sejumlah komputer yang biasanya terkoneksi dalam *Local Area Network* (LAN), yang bekerja seakan-akan mereka adalah sebuah komputer. Tujuan dari *computer cluster* ini adalah untuk meningkatkan performa komputer dengan biaya yang relatif murah. *Computer cluster* yang akan digunakan adalah model Beowulf [12].

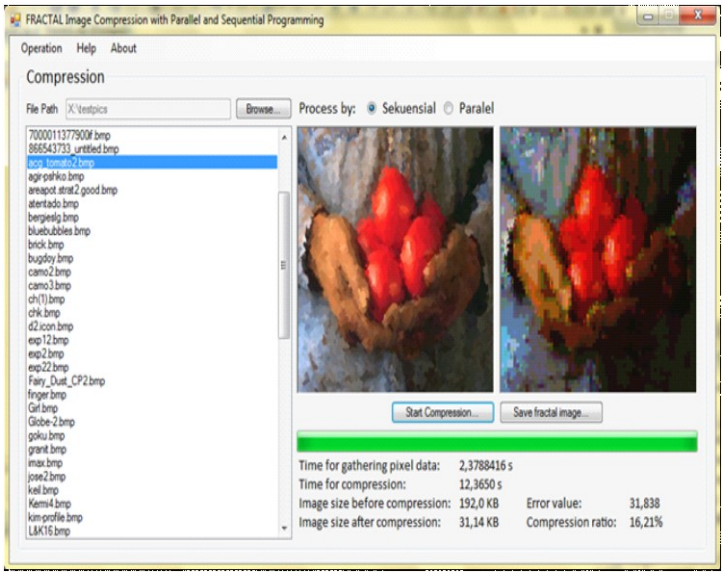
Dengan nama yang berasal dari cerita kepahlawanan Inggris zaman dahulu, *Computer Cluster* dengan model Beowulf ini menggunakan komputer-komputer identik yang relatif murah. Komputer-komputer ini saling terhubung dalam sebuah LAN dan di dalamnya terdapat program-program yang memungkinkan untuk membagi-bagi proses diantara komputer-komputer tersebut. Pada umumnya, model pemrosesan paralel yang digunakan adalah MPI (*Message Passing Interface*). *Cluster* komputer ini terdiri dari tiga unit komputer dengan spesifikasi yang sama yaitu, prosesor Intel Pentium IV 1.66 GHz, 1GB RAM, dan NIC Broadband *gigabit Ethernet.*

Spesifikasi program kompresi Fraktal yang pertama adalah modul *form* kompresi di mana *form* kompresi ini digunakan saat akan melakukan kompresi citra. Pada *form* ini terdapat tampilan citra awal, tampilan citra setelah kompresi, *radio button sequential*-paralel, *button open*, *button* kompresi, *button save*, dan *button back*. *Button open* digunakan untuk membuka *file* citra yang akan dikompresi. Radio *button sequential*-paralel untuk memilih apakah proses akan dilakukan secara *sequential* atau paralel. *Button* kompresi digunakan untuk memulai proses kompresi setelah melakukan pengaturan. *Button save* digunakan untuk menyimpan *file* citra yang telah dikompresi. *Button back* digunakan untuk kembali ke menu utama. Tampilan *form* dapat dilihat pada gambar 4.

Spesifikasi kedua yaitu modul *form* dekompresi di mana *form* dekompresi ini digunakan saat akan melakukan kompresi citra. Pada *form* ini terdapat tampilan citra awal, tampilan citra setelah dekompresi, *radio button sequential*-paralel, *button open*, *button* dekompresi, *button save*, dan *button back. Button open* digunakan untuk membuka *file* citra yang akan didekompresi. Radio *button sequential*-paralel untuk memilih apakah proses akan dilakukan secara *sequential* atau paralel. *Button* dekompresi digunakan untuk memulai proses dekompresi. *Button save* digunakan untuk menyimpan *file* citra yang telah didekompresi. *Button back* digunakan untuk kembali ke menu utama. Tampilan *form* dapat dilihat pada gambar 5.



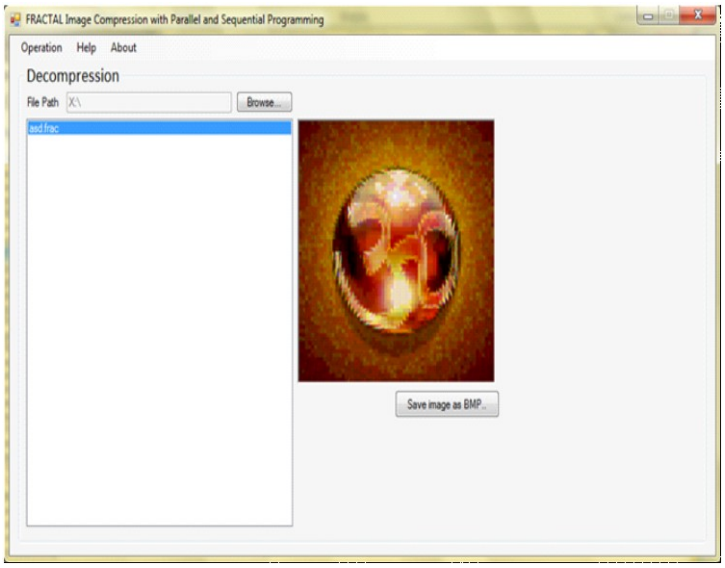
Gambar 3. Pemetaan proses kompresi Fraktal paralel.



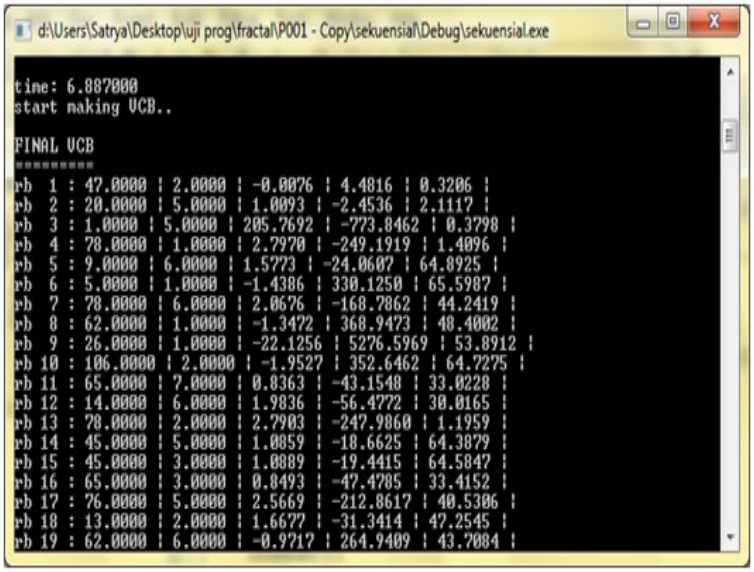
Gambar 4. Tampilan *form* untuk modul kompresi.

Pengujian program aplikasi ini dilakukan secara empat tahap. Tahap pertama adalah pengujian prosedur pengambilan data *piksel* dari tiap elemen warna menggunakan Visual Basic (VB) dan disimpan ke dalam tiga *file* teks, red.txt, green.txt, blue.txt. Tahap kedua adalah pengujian proses kompresi citra secara *sequential* dari membaca nilai-nilai *piksel* tiap elemen warna dari *file* teks hingga pembuatan *file* *.FRAC pada C. Tahap ketiga adalah pengujian proses kompresi sampai dengan pembuatan *file* *.FRAC secara paralel di C. Tahap terakhir adalah pengujian dengan 57 citra sampel untuk mendapatkan hasil citra *output*.

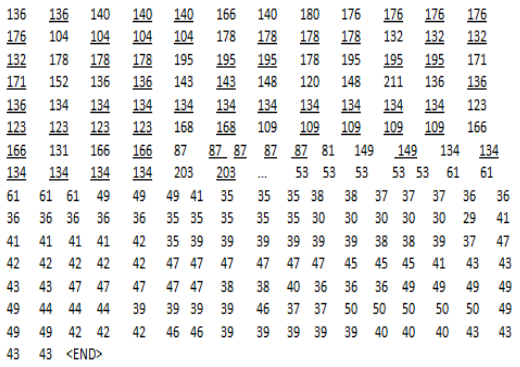
Pengujian prosedur pengambilan data *piksel* pada setiap elemen citra *input* dilakukan pada VB, dengan citra *input* berupa citra RGB berformat BMP dengan resolusi 256x256. Prosedur ini menghasilkan *output* berupa tiga *file* teks, red.txt yang berisi data *piksel* elemen warna merah, green.txt yang berisi data *piksel* elemen warna hijau, dan blue.txt yang berisi data *piksel* elemen warna biru. Ketiga *file* tersebut disimpan pada direktori X:\. Contoh isi *file* red.txt dapat dilihat pada gambar 6.



Gambar 5. Tampilan *form* untuk modul dekompresi.



Gambar 7. Tampilan virtual *codebook*



Gambar 6. Isi *file* Red.txt.

Pengujian proses kompresi citra secara *sequential* dilakukan pada C++ dengan *input* berupa tiga *file* teks yang berisi data *piksel* tiap elemen warna hasil dari prosedur pengambilan data *piksel* sebelumnya. Proses ini menghasilkan *output* berupa tampilan virtual *codebook* final pada *command prompt, file* *.FRAC yang disimpan pada direktori X:\. Tampilan virtual *codebook* yang dihasilkan dapat dilihat pada gambar 7.

Pengujian proses kompresi citra secara paralel dilakukan pada C dengan input berupa tiga file teks yang berisi data piksel tiap elemen warna hasil dari prosedur pengambilan data piksel sebelumnya. Proses ini berhasil melakukan komunikasi dengan dua unit komputer lainnya yang terlibat dalam proses paralel. Proses ini juga berhasil menghasilkan output berupa tampilan virtual codebook final pada command prompt, file *.FRAC yang disimpan pada direktori X:\.

Citra yang diproses dalam pengujian terhadap file citra berjumlah 57 citra BMP 24 bit. Pada citra-citra yang diuji tersebut akan dilakukan pendataan terhadap nilai rasio kompresi, waktu proses sequential, waktu komputasi proses paralel, waktu komunikasi proses paralel, rasio waktu komputasi atau komunikasi, total waktu proses paralel, nilai speed-up, dan nilai efisiensi tiap prosesor.

Pengujian proses kompresi citra secara paralel dilakukan pada C dengan *input* berupa tiga *file* teks yang berisi data *piksel* tiap elemen warna hasil dari prosedur pengambilan data *piksel* sebelumnya. Proses ini berhasil melakukan komunikasi dengan dua unit komputer lainnya yang terlibat dalam proses paralel. Proses ini juga berhasil menghasilkan *output* berupa tampilan virtual *codebook* final pada *command prompt, file* *.FRAC yang disimpan pada direktori X:\.

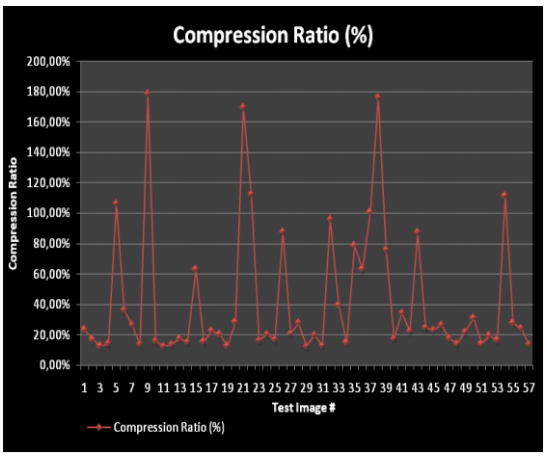
Citra yang diproses dalam pengujian terhadap *file* citra berjumlah 57 citra BMP 24 bit. Pada citra-citra yang diuji tersebut akan dilakukan pendataan terhadap nilai rasio kompresi, waktu proses *sequential*, waktu komputasi proses paralel, waktu komunikasi proses paralel, rasio waktu komputasi/komunikasi, total waktu proses paralel, nilai *speed-up*, dan nilai efisiensi tiap prosesor.

Dari gambar 8 yang menampilkan rasio kompresi dari 57 citra 24-bit BMP, hasil pengujian menunjukkan bahwa rasio kompresi rata-rata dengan menggunakan metode fraktal adalah 41.79%. Dapat dilihat juga dari grafik tersebut, Rasio kompresi yang dihasilkan sangat bervariasi. Hal ini terjadi karena penggunaan nilai CV, yang sangat dipengaruhi oleh tingkat

kedetilan suatu citra. Semakin kecil nilai CV yang digunakan semakin banyak cluster yang terbentuk dan semakin kecil pula rasio kompresi citra tersebut.

Sehubungan dengan waktu eksekusi dari pengujian beberapa citra uji, hasil yang didapat cukup signifikan dengan nilai *speed-up* sebesar 1.69. Hasil ini dapat dilihat dengan jelas pada gambar 9 dan gambar 10. Lebih lanjut, dari hasil pengujian juga didapatkan bahwa setiap prosesor digunakan lebih dari separuh total komputasi. Hal ini ditunjukkan dari nilai rata-rata efisiensi prosesor sebesar 56.34%.

Dalam pemrosesan paralel, *granularity* adalah suatu rasio perbandingan antara komunikasi dan komputasi dalam sebuah algoritma paralel. Dua model dari *granularity* ini adalah *fine-grain* dan *coarse-grain*. Pada model *fine-grain* rasio komputasi terhadap komunikasi cukup kecil. Model ini memungkinkan untuk mengoptimalkan *load balancing*, tetapi sulit untuk meningkatkan performa karena *overhead* komunikasi yang tinggi. Pada model *coarse-grain*, rasio komputasi terhadap komunikasi relatif besar. Kemungkinan untuk meningkatkan performa lebih besar, tetapi lebih sulit untuk mengimplementasikan *load balancing*. *Granularity* dari algoritma paralel ini adalah *coarse-grain*, di mana rasio komputasi per komunikasi dari kebanyakan citra uji menunjukkan nilai di atas 1. Grafik yang menunjukkan rasio komputasi per komunikasi dapat dilihat pada gambar 11.



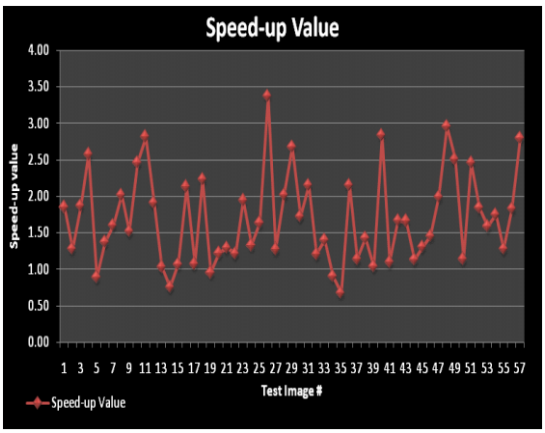
Gambar 8. Rasio kompresi dari 57 citra 24-bit BMP.

Pada beberapa citra uji rasio memiliki nilai di bawah 1. Hal tersebut terjadi karena tingkat dependensi yang tingga saat penulisan *file* FRAC, di mana proses penulisan *file* harus dilakukan berurutan dari proses 0 ke proses 2. Untuk citra dengan watku proses pada tiap elemen warna yg hampir sama, waktu komunikasi akan menjadi lebih singkat. Tetapi waktu proses yang lama pada elemen warna yang diproses setelah elemen awal (elemen warna hijau atau biru) akan menghasilkan waktu komunikasi yang lebih lama.

4. Kesimpulan

Berdasarkan hasil pengujian dari bagian sebelumnya, peneliti dapat menjabarkan beberapa kesimpulan yaitu algoritma paralel yang digunakan berhasil mempersingkat waktu kompresi dengan nilai *speed-up* rata-rata sebesar 1.69 dan tingkat efisiensi sebesar 56.34%. Algoritma paralel yang digunakan memiliki *granularity coarse-grain*, di mana total waktu komputasi lebih besar daripada total waktu komunikasi.

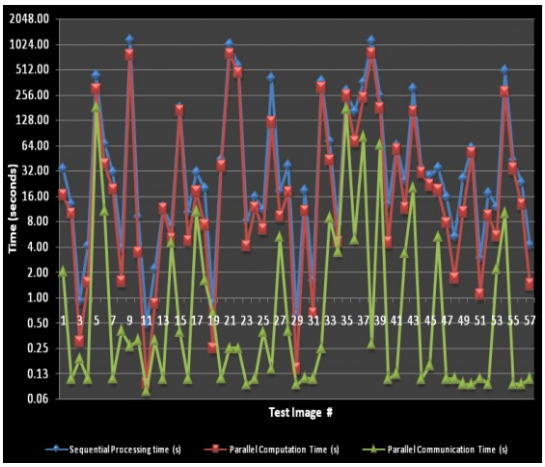
Citra dengan tingkat kompleksitas yang lebih tinggi berpotensi meningkatkan jumlah *cluster*, sehingga membuat waktu kompresi menjadi lebih lambat dan mengurangi rasio kompresi. Dependensi saat menulis *file* FRAC membuat citra dengan jumlah *cluster* yang lebih sedikit pada elemen warna awal (elemen warna yang lebih dulu ditulis ke *file*) memiliki waktu komunikasi yang lebih lama.



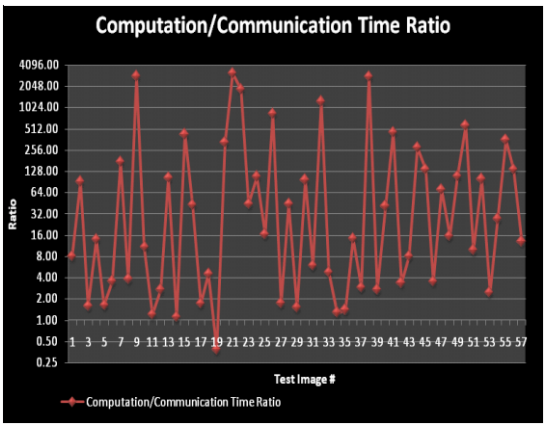
Gambar 9. Nilai *speed-up* dari proses kompresi algoritma Fraktal dengan 57 citra 24-bit BMP.

Untuk penelitian selanjutnya, eksperimen yang akan dilakukan difokuskan pada beberapa hal antara lain, melakukan pengujian dengan menggunakan partisi blok citra sebesar 2x2 untuk meningkatkan kualitas kompresi dan membuat waktu pemrosesan lebih lambat, sehingga lebih baik jika menerapkan algoritma paralel. Meningkatkan nilai *speed-up* pada proses paralel dengan meneliti kemungkinan memaralelkan langkah-langkah dari proses kompresi Fraktal

mengurangi sampai menghilangkan dependensi saat menulis *file* FRAC pada algoritma paralel.



Gambar 10. Perbandingan waktu *sequential*, waktu komputasi paralel, dan waktu komunikasi paralel dari 57 citra 24-bit BP.



Gambar 11. *Granularity* pada algoritma paralel dari 57 citra 24-bit BMP.

Referensi

[1] Uni-Oldenburg, Fractal Image Compression, Uni-Oldenburg http://www.rasip.fer.hr/researchs/compress/algorithms/adv/fraccomp/index.html, retrieved August 20, 2008.

[2] G. Ong & L. Fan, "An Efficient Parallel Algorithm for Hexagonal-based Fractal Image compression," *International Journal of Computer Mathematics*, vol. 84, pp. 203-218, 2007.

[3] H. Peng, M. Wang, & C.H. Lai, "Design of Parallel Algorithms for Fractal Video Compression," *International Journal of Computer Mathematics*, vol. 84, pp. 193-202, 2007.

[4] R.C. González & R.E. Woods, *Digital Image Processing*, 3rd ed., Prentice Hall, New Jersey, 2008.

[5] P.J. Burt & E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. on Communications*, vol. 31, pp. 532–540, 1983.

[6] H.A. Anton & C. Rorres, *Elementary Linear Algebra Applications Version*, 8th ed., John Wiley & Sons, Inc., United States, 2000.

[7] Texas Instrument, An Introduction to Fractal Image Compression, Texas Instrument Incorporated, http://focus.ti.com/lit/an/bpra065/bpra065.pdf, 1997, retrieved August 19, 2008.

[8] A.J. Hobson, Statistic 3, University of East Anglia, http://www.mth.uea.ac.uk/jtm/18/Lec18p3.pdf, retrieved August 23, 2008.

[9] Math Central, Quandaries and Queries, Math Central, http://mathcentral.uregina.ca/QQ/database/QQ.09.05/Jan1.html, retrieved August 23, 2008.

[10] Alexandria University, "A Novel Secure Image Coding Scheme Using Fractal Transformation", longwood, http://longwood.cs.ucf.edu/~ahossam/research/paper/URSI98.pdf, retrieved August 20, 2008.

[11] Argonne National Laboratory, MPICH for Windows NT, Mathematics and Computer Science Division, http://www.mcs.anl.gov/mpi/mpich/download.html, retrieved April 12, 2007.

[12] R.G. Brown, Engineering a Beowulf-style Compute Cluster, Duke University Physics Department, http://www.phy.duke.edu/~rgb/Beowulf/beowulf_book/beowulf_book/index.html, 2004, retrieved May 8, 2008.